



UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/608,330	06/27/2003	Eric Gouriou	200206152-1	7978
<div>22879 7590 05/01/2007</div> <div>HEWLETT PACKARD COMPANY P O BOX 272400, 3404 E. HARMONY ROAD INTELLECTUAL PROPERTY ADMINISTRATION FORT COLLINS, CO 80527-2400</div>				
			<div>EXAMINER</div> <div>FRANCIS, MARK P</div>	
			<div>ART UNIT</div> <div>2193</div>	<div>PAPER NUMBER</div>
			<div>MAIL DATE</div> <div>05/01/2007</div>	<div>DELIVERY MODE</div> <div>PAPER</div>

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

**Advisory Action
Before the Filing of an Appeal Brief**

Application No.

10/608,330

Applicant(s)

GOURIOU ET AL.

Examiner

Mark P. Francis

Art Unit

2193

--The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

THE REPLY FILED 05 February 2007 FAILS TO PLACE THIS APPLICATION IN CONDITION FOR ALLOWANCE.

1. ☒ The reply was filed after a final rejection, but prior to or on the same day as filing a Notice of Appeal. To avoid abandonment of this application, applicant must timely file one of the following replies: (1) an amendment, affidavit, or other evidence, which places the application in condition for allowance; (2) a Notice of Appeal (with appeal fee) in compliance with 37 CFR 41.31; or (3) a Request for Continued Examination (RCE) in compliance with 37 CFR 1.114. The reply must be filed within one of the following time periods:

- a) ☒ The period for reply expires 3 months from the mailing date of the final rejection.
b) ☐ The period for reply expires on: (1) the mailing date of this Advisory Action, or (2) the date set forth in the final rejection, whichever is later. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of the final rejection.

Examiner Note: If box 1 is checked, check either box (a) or (b). ONLY CHECK BOX (b) WHEN THE FIRST REPLY WAS FILED WITHIN TWO MONTHS OF THE FINAL REJECTION. See MPEP 706.07(f).

Extensions of time may be obtained under 37 CFR 1.136(a). The date on which the petition under 37 CFR 1.136(a) and the appropriate extension fee have been filed is the date for purposes of determining the period of extension and the corresponding amount of the fee. The appropriate extension fee under 37 CFR 1.17(a) is calculated from: (1) the expiration date of the shortened statutory period for reply originally set in the final Office action; or (2) as set forth in (b) above, if checked. Any reply received by the Office later than three months after the mailing date of the final rejection, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

NOTICE OF APPEAL

2. ☐ The Notice of Appeal was filed on _____. A brief in compliance with 37 CFR 41.37 must be filed within two months of the date of filing the Notice of Appeal (37 CFR 41.37(a)), or any extension thereof (37 CFR 41.37(e)), to avoid dismissal of the appeal. Since a Notice of Appeal has been filed, any reply must be filed within the time period set forth in 37 CFR 41.37(a).

AMENDMENTS

3. ☐ The proposed amendment(s) filed after a final rejection, but prior to the date of filing a brief, will not be entered because
(a) ☐ They raise new issues that would require further consideration and/or search (see NOTE below);
(b) ☐ They raise the issue of new matter (see NOTE below);
(c) ☐ They are not deemed to place the application in better form for appeal by materially reducing or simplifying the issues for appeal; and/or
(d) ☐ They proposed additional claims without canceling a corresponding number of finally rejected claims.

NOTE: _____. (See 37 CFR 1.116 and 41.33(a)).


4. ☐ The amendments are not in compliance with 37 CFR 1.121. See attached Notice of Non-Compliant Amendment (PTOL-324).
5. ☐ Applicant's reply has overcome the following rejection(s): _____.
6. ☐ Newly proposed or amended claim(s) _____ would be allowable if submitted in a separate, timely filed amendment canceling the non-allowable claim(s).
7. ☒ For purposes of appeal, the proposed amendment(s): a) ☐ will not be entered, or b) ☒ will be entered and an explanation of how the new or amended claims would be rejected is provided below or appended.
The status of the claim(s) is (or will be) as follows:
Claim(s) allowed: _____.
Claim(s) objected to: _____.
Claim(s) rejected: 10, 11, 13, 29, 30, 32, 35 and 37-39.
Claim(s) withdrawn from consideration: _____.

AFFIDAVIT OR OTHER EVIDENCE

8. ☐ The affidavit or other evidence filed after a final action, but before or on the date of filing a Notice of Appeal will not be entered because applicant failed to provide a showing of good and sufficient reasons why the affidavit or other evidence is necessary and was not earlier presented. See 37 CFR 1.116(e).
9. ☐ The affidavit or other evidence filed after the date of filing a Notice of Appeal, but prior to the date of filing a brief, will not be entered because the affidavit or other evidence failed to overcome all rejections under appeal and/or appellant fails to provide a showing of a good and sufficient reasons why it is necessary and was not earlier presented. See 37 CFR 41.33(d)(1).
10. ☐ The affidavit or other evidence is entered. An explanation of the status of the claims after entry is below or attached.

REQUEST FOR RECONSIDERATION/OTHER

11. ☒ The request for reconsideration has been considered but does NOT place the application in condition for allowance because:
See Continuation Sheet.
12. ☐ Note the attached Information Disclosure Statement(s). (PTO/SB/08) Paper No(s). _____
13. ☐ Other: _____.


MENG-AT T. AN
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 210

Continuation of 11. does NOT place the application in condition for allowance because: Applicant has taken into consideration all of Applicants' arguments but maintains his grounds of rejection. Following is the Examiner's response.

The Office acknowledges that a typographical error was made by the Examiner regarding claim 29 and it will be removed from the list of rejected claims in the next official paper.

The 101 Non-Statutory rejection of claim 35 is upheld by the Examiner. Even if the Applicants' position is true that a suite of software could rightly be described as a system. Claim 35 would still be Non-Statutory for the reasons that the body of the claim comprises a parent process and a process monitor which are considered to be software means that can be implemented using software means only, (i.e. computer programs per se) and do not require the use of hardware or produce a tangible result of a practical application.

With respect to claims 10, 11, 13, and 37, Applicant essentially argues that Bennett et. al does not teach or disclose a method for controlling the execution of a child process.

In reply, the Examiner disagrees, Note Col 7:13-24, it is here that Bennett teaches that the OS process control files provide a memory area for each process that can be opened, read, and written to by other processes. Bennett teaches that each control file can include several control messages, which are used to govern the process. In addition, Bennett teaches that the messages include scripts for handling control information regarding the process and the messages may include monitoring threads, call traces, or commands to suspend or continue processing. Therefore, Bennett does teach a method for controlling the execution of a child process.

With respect to claims 10, 11, 13, and 37, Applicant essentially argues that Bennett does not teach instrumenting a parent process.

In response, the Examiner differs, Note Col 6:30-42, it is here that Bennett discloses that the OS fork routine invoked by the OS to start a new process. The new process or child process is an exact copy of the calling process, or parent process and the child process inherits most of its attributes from the parent process, but it receives its own process ID, entries within the process control data structure, associated memory allocations, and a few other unique attributes. Bennett discloses that the OS fork routine is used by the OS to start all new processes and that some operating systems include multiple fork routines with associated system calls. Since instrumenting a child process in Bennett's invention is done by inheriting most of its attributes from the parent process, it is inherent that Bennett teaches instrumenting a parent process which is the OS fork routine 224 that is invoked by the OS to start a new process. Thus, Bennett does teach instrumenting a parent process.

In addition, regarding claims 10, 11, 13, and 37, Applicant essentially argues that Bennett does not teach before a vfork system call is executed, receiving with the process monitor indicia from the parent process that a vfork system call will be executed by the parent process.

In reply, the Examiner disagrees, Note, Col 6:30-42, it is here that Bennett discloses that once installed, the fault detection driver will accumulate process IDs (monitor indicia) for each new process as that new process is invoked via the OS fork routine or in this case the Parent process. In addition, Bennett teaches that some OS include multiple fork routines with associated system calls for example Solaris includes fork(), vfork(), and fork() 1 system calls for accessing three variations on the OS fork routine (Parent Process). Thus, Bennett does teach before a vfork system call is executed, receiving with the process monitor indicia from the parent process that a vfork system call will be executed by the parent process.

Also, the Applicant argues that Bennett does not teach an indication is provided to any process monitor.

In response, the Examiner differs, Note Col 2: 45-53, it is here that Bennett teaches that the fault monitor signals interruptions to an event router which may, in turn, signal the interruption and provide key information to fault handling resource, such as a fault administration system that is located within or outside the UNIX environment. Therefore, Bennett does teach an indication is provided to a process monitor.

Regarding claims 10, 11, 13, and 37, Applicant essentially argues that Bennett does not teach the action of suspending execution of the parent process.

In reply, the Examiner differs, Note Col 10:25-35, it is here that Bennett teaches that a monitor suspension control message is also stored in the process control file. The monitor suspension control message suspends the monitoring thread until the monitored process is interrupted.

Fifth, Applicant argues that Bennett does not teach or disclose extracting with the process monitor a process identifier from the indicia, the process identifier identifying a child process to be generated by the parent process when the parent process executes the vfork system call.

In reply, the Examiner disagrees, Note Col 8:33-45, it is here that Bennett discloses that the fault detection driver returns all process IDs that it is presently storing to the fault monitor to determine for which processes it has already started a monitoring thread and for which processes a new thread should be started. Therefore, for each process that the fault monitor determines that a new thread should be started, an open read call or v-fork call is sent to the fault detection driver as soon as it has returned and handled the prior batch of process IDs. The new process ID are returned to the fault monitor in the next read call and generates a monitoring thread corresponding to the new process ID of the calling process. In addition, the Examiner Note: Col 6:30-45, it is here that Bennett discloses that the OS fork routine is used by the OS to start all new processes and that some operating systems include multiple fork routines such as

vfork() with associated system calls. Therefore, Bennett does teach extracting with the process monitor a process identifier from the indicia, the process identifier identifying a child process to be generated by the parent process when the parent process executes the vfork system call.

Sixth, Applicant argues that Bennett does not teach setting with the process monitor a process monitor thread to observe trace events generated by the child process.

In response, the Examiner disagrees, Note Col 7:40-64, it is here that Bennett teaches the fault monitor interacts with the OS process control file and the OS status file associated with the process n to establish monitoring conditions, identify interruptions,(trace events) and analyze those interruptions.(trace events) Bennett also teaches that the fault monitor initiates fault monitoring by issuing an open call to the fault detection driver. After the fault detection driver receives the open call, it starts monitoring for new processes.

Next, Applicant essentially argues that Bennett does not teach resuming execution of the parent process to enable the parent process to execute the vfork system call.

In reply, the Examiner differs, Note Col 10:55-64, it is here that Bennett teaches that the fault monitor may evaluate the signal, status, and preferences and can make a determination that the signal can be handled by the monitored process. The fault monitor writes a control message back to the process control file(parent process) to cause it to continue processing and re-writes the monitor suspending control message and then monitoring continues as it had prior to the interruption. Therefore, Bennett does teach resuming execution of the parent process to enable the parent process to execute vfork system call.

With respect to claims 35,38,and 39, Applicant argues that Bennett fails to teach a parent process configured to generate a pre-fork event that contains a process identifier of a child process that will be spawned from the parent process.

In reply, the Examiner disagrees, Note Col 8:33-45, it is here that Bennett discloses that the fault detection driver returns all process IDs that it is presently storing to the fault monitor to determine for which processes it has already started a monitoring thread and for which processes a new thread should be started. Therefore, for each process that the fault monitor determines that a new thread should be started, an open read call or v-fork call is sent to the fault detection driver as soon as it has returned and handled the prior batch of process IDs. The new process ID are returned to the fault monitor in the next read call and generates a monitoring thread corresponding to the new process ID of the calling process. In addition, the Examiner Note: Col 6:30-45, it is here that Bennett discloses that the OS fork routine is used by the OS to start all new processes and that some operating systems include multiple fork routines such as vfork() with associated system calls. Therefore, Bennett does teach a parent process configured to generate a pre-fork event that contains a process identifier of a child process that will be spawned from the parent process.

With respect to claims 35, 38, and 39, Applicant essentially argues that Bennett fails to disclose a process monitor configured to receive the pre-fork event and process identifier before the vfork system call is executed, suspend execution of the parent process, or generate a process monitor thread that enables observation of trace events generated by the child process.

In response, the Examiner differs, Note Col 6:30-42, it is here that Bennett discloses that the OS fork routine invoked by the OS to start a new process. The new process or child process is an exact copy of the calling process, or parent process and the child process inherits most of its attributes from the parent process, but it receives its own process ID, entries within the process control data structure, associated memory allocations, and a few other unique attributes. Bennett discloses that the OS fork routine is used by the OS to start all new processes and that some operating systems include multiple fork routines with associated system calls. Since instrumenting a child process in Bennett's invention is done by inheriting most of its attributes from the parent process, it is inherent that Bennett teaches instrumenting a parent process which is the OS fork routine 224 that is invoked by by the OS to start a new process. Thus, Bennett does teach instrumenting a parent process. In addition, regarding claims 35,38, and 39, Applicant essentially argues that Bennett does not teach before a vfork system call is executed, receiving with the process monitor indicia from the parent process that a vfork system call will be executed by the parent process. In reply, the Examiner disagrees Note, Col 6:30-42, it is here that Bennett discloses that once installed, the fault detection driver will accumulate process IDs(monitor indicia) for each new process as that new process is invoked via the OS fork routine or in this case the Parent process. In addition, Bennett teaches that some OS include multiple fork routines with associated system calls for example Solaris includes fork(), vfork(), and fork() 1 system calls for accessing three variations on the OS fork routine(Parent Process). Thus, Bennett does teach before a vfork system call is executed, receiving with the process monitor indicia from the parent process that a vfork system call will be executed by the parent process.